3L-3

可変構造多重処理データベースマシンのシステム構成 ＝ソートモジュール＝

喜連川優　鈴木重信　田中英彦　元岡達
東京大学工学部

§0 はじめに

可変構造多重処理データベースマシンは[1] Cellular Logic Type のDBMをより発展させたMIMD型マシンである。(図1)多重チャネルリングバスにより、多くのプロセッシングモジュール(PM)群とメモリモジュール(MM)群との間に柔軟な結合関係を生成し、処理負荷に応じた動的プロセッサ割付けを可能としている。又メモリからプロセッサ群へは、データブロードキャストを行ないコンフリクトのない高次の並列処理を行なう。今回、更に処理の高速化を計る為、ソートモジュール(SM)の検討を行なったので報告する。

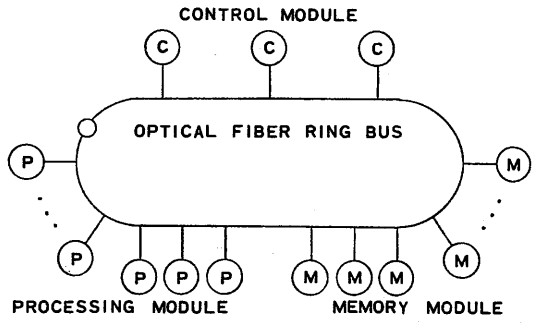

FIG.1 ARCHITECTURE OF DATA MANIPULATION UNIT

§1 ソートモジュールの設計指針

1)入力データストリームに同期したソート ---- MMは割当てられた当該チャネルに対しタプルを連続的に出力する。SMはメモリからのデータ入力に遅れる事なくソートを行ない処理と入力をオーバラップさせ、最後のデータ入力終了と共に、すぐに、ソート出力が得られる構成が好ましい。

2)タプル長に対する柔軟性 ---- タプルデータをコード化し、対象データを一定長に制限する事は設計を容易にするが、ここではより一般的にタプル長の変更に対しても、柔軟に対処できる様にする。但し、簡単の為1リレーションのストリーム内ではタプル長は固定とする。(ドメイン選択バブルメモリを仮定)

3)半導体RAM技術の進歩への期待 ----LSI化を考える際 Systolic Array[2] の如くロジックとメモリセルの一体化が自然であるが、より一般的な制御構造(Programability)を入れ易くする為、メモリチップとプロセッサチップに分離することとし、$\log_k N$ (N: ソートタプル数, K: K-wayソート) 台のプロセッサと大容量半導体RAMバンクによる構成とした。

§2 ソートモジュールの構成

2.1 ソートアルゴリズム ----アルゴリズムは3),4)によって既に知られているパイプラインパラレルマージソートであり、図2に示される如くプロセッサは各ステージで等しく、メモリは2-wayの場合 第iステージでは 2^i ヶのタプル分の容量をもつ。iステージ目のプロセッサは $i-1$ ステージ目のプロセッサから与えられる 2^i ヶのタプルからなる2本のソートされたストリームをマージして 2^{i+1} ヶのタプルからなる1本のストリームとして $i+1$ 番目のプロセッサへ出力する。前のストリームに対するマージと次のストリームの入力をオーバラップさせパイプライン処理する事により、$\log_2 N$ 台のプロセッサにより $O(N)$ の時間でソート出来る。

2.2 メモリ管理方式 ---- iステージ目のプロセッサは 2^i ヶのタプルからなるストリームを入力した後、次のストリームに対し、1タプル入力しては比較後1タプル出力する為常に有効なメモリ容量は一定であり、2タプル分あればよい事になる。ところが、ストリームトップをとる順序はランダムであり、空いた所に入力していくのではそのソートされた順序を維持出来ない。これに対し、ここでは次の様な方式を検討した。

a) Double Memory Method ----最も単純な管理手法は2倍の容量を用意し、各々独立したFIFOとして、2つのストリームに割当てる事である。しかし、i段目では、2^i ヶのタプルを入力する為、iが大きくなると、この無駄は無視出来ない。

b) Pointer Method ---- ポインタによって、タプル間のリンクをとって行く手法

c) Block Division Method ---- a)とb)の中間策であり、メモリを固定長のブロックに分割し、ブロック内ではデータはソートされており、ブロック間の前後関係を別の構造としてもたせる。今、当該ステージでその必要なメモリ域をlケのブロックに分割するとl+2ケのブロック分のメモリを用意する事によって1ブロックのデータ入力後必ず1つの空ブロックが生成されるという点に着目し、ブロック単位のメモリ管理を行なう。

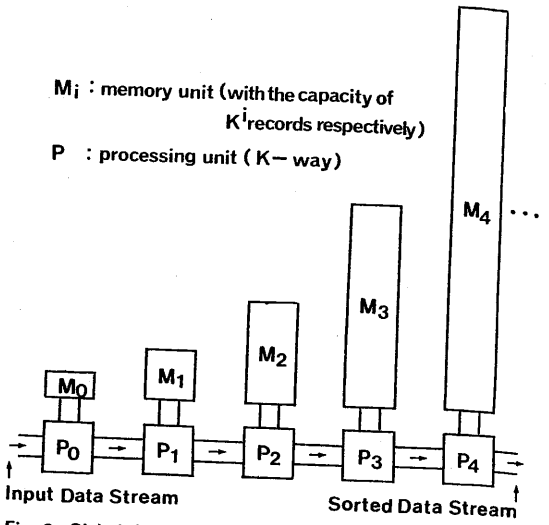

Fig.2 Global Architecture Of Stream Driven Sorter

2.3 レコード長による管理方式の選択

A) 短いデータのソート---- キー部のみを高速にソートする場合、又それがエンコードされている様な場合には、1タプルはかなり短くなる（数バイト）。従ってb)の手法を用いるとポインタ部だけでも2~3バイト必要となり無視出来ない。この場合はc)の手法を採用する事により、効率化が図れる。最適ブロック分割数を各段で決定する事が出来、必要なメモリ容量は1~2%増に過ぎない。

B) 長いデータのソート---- 一般的にデータの格納はレコード単位に行なわれフィールド分割は行なっていない。従ってデータ出力レートは一定であり、パイプラインソータでは時間的ロスがない為、キー部と非キー部の分離には意味がなく、フルレコードを直接ソートした方が便利な事が多い。この様な場合には、ポインタの占める割合はわずかであり、制御の容易さからb)の手法が好ましい。

2.4 ポインタ方式でのメモリ効率 ---- 2.3のB)ではリレーション間でタプル長が変化しても、メモリ効率を高くし、成べく多くのタプルをソート出来る事が望ましい。K-wayソータに於いて、(Kが大きいと、ステージ数が少なくなり、メモリバンクの細分化が僅かですむ。) タプル長をLとして、各ステージのメモリ容量を決定したとする。この時、実際のレコード長Xに対し、メモリ効率は $\eta = [K \cdot L/X] \cdot X/K \cdot L$ で表わされ、$L < X < K \cdot L$ で最低0.5となる事がわかる。これに関しては最初の数段のメモリを $K-1/K+1$ だけ増加し、最初のプロセッサがストリーム長を調整する事にすれば、K=8~16の時平均97~98%の効率が得られる。

§3. 関係演算の処理

当該リレーションを占める複数台のMMからのデータ出力はハッシュを施されて、互いに独立した幾つかのバケットに分割される。SMは各バケットに割付けられ、データの入力に従ってソートを行なう。データの分配が終了すると直ちにソート出力が可能であり、各モジュールは独立に処理を開始する。SMはソータと関係演算処理部からなり、Projectionでは、$O(n)$で重複除去を行なえ、又、Divisionでは Divisor の Cardinarity が大きい場合一部で割り飾い落としをすると共に、商の可能性のあるタプルを再びソータの入力とする事で処理出来る。以上の如く、データがソートされている場合には比較的負荷の重い Join Projection Division も高速に処理出来る事は明らかである。

§4. むすび

現在ソータのLSI化の設計を行なっている。今後SMの設計、システムレベルの制御手法の研究を進める予定である。

参考文献 1. 喜連川他 信学技報 EC 80-51, 1980
2. H.T. Kung Advances in Computers vol 19 1980 P65
3. Shimon Even CACM April Vol 17 No 4 1974 P202
4. S. Todd IBM J. RES. DEVELOP Vol. 22 No 5 1978 P509